

ATLAS
MANUAL DE USUARIO
COMPONENTE LISTA DE VALORES

Versión 1.9

Arquitectura de Software



Agencia de Informática y Comunicaciones de la Comunidad de Madrid



Hoja de Control

Título	Manual de usuario Componente Lista de valores		
Documento de Referencia	NORMATIVA ATLAS		
Responsable	Arquitectura de Software		
Versión	1.9	**Fecha Versión**	11/07/2013

Registro de Cambios

Versión	Causa del Cambio	Responsable del Cambio	Fecha del Cambio
1.0	Versión inicial del documento	Área de Integración y Arquitectura de Aplicaciones	26/05/2010
1.1	Modificaciones sobre la versión 1.0: - **4.1.2 Paso 2: Declaración en el bean de respaldo de la página de un objeto de tipo atlas.componente.bo.ListaValores e inicialización del mismo:** Utilizar una clase dentro de un bean en lugar de un bean en el contexto de JSF para cada lista de valores. - **4.1.4 Paso 4: Inserción en la página de la etiqueta de Atlas:** Eliminado atributo regsLimit y creado nuevos atributos ajax y tridentIVEngineSelectBehavior para la lista y aliasTabla para las columnas.	Área de Integración y Arquitectura de Aplicaciones	03/09/2010
1.2	Modificaciones: - **4.1.4 Paso 4: Inserción en la página de la etiqueta de Atlas:** Modificado atributo ajax y añadido atributo reRender de la lista de valores. Añadido atributo for de columnaListaValores. - **4.1.5 Funcionamiento con ajax.** - Añadidas preguntar al FAQ sobre el funcionamiento con ajax.	Área de Integración y Arquitectura de Aplicaciones	19/10/2010

Versión	Causa del Cambio	Responsable del Cambio	Fecha del Cambio
1.3	Modificaciones: - **2. Descripción:** Actualizada captura del componente. - **4.1.4 Paso 4: Inserción en la página de la etiqueta de Atlas:** Actualizados atributos.	Área de Integración y Arquitectura de Aplicaciones	08/02/2011
1.4	- **4.1.4 Paso 4: Inserción en la página de la etiqueta de Atlas:Lista de valores:** Añadido id a la lista de valores, es obligatorio como se especifica más adelante.. - Se modifica el nombre del Area	Área de Aplicaciones Especiales y Arquitectura de Software	01/09/2011
1.5	Modificación general del documento por la migración del componente a JSF 2.	Área de Aplicaciones Especiales y Arquitectura de Software	15/02/2012
1.6	Modificacion para utilizar un provider externo en lugar de la consulta en el fichero queries.properties. Esto permite usar la lista de valores sin conexión a base de datos. Modificación del documento para los nuevos modos de renderizado del componente (con/sin campo de texto y modo suggestion).	Área de Aplicaciones Especiales y Arquitectura de Software	13/06/2012
1.7	**4.3 Atributos del componente**: añadido atributo filtrarAcentos.	Área de Aplicaciones Especiales y Arquitectura de Software	04/10/2012
1.8	**4.3 Atributos del componente**: ampliada la descripción de la propiedad autosized y añadido nuevo atributo suggestionSize.	Área de Aplicaciones Especiales y Arquitectura de Software	17/01/2013
1.9	**4.3 Atributos del componente**: añadidos atributos inputSize, inputWidth e inputClass.	Arquitectura de Software	16/04/2013

Índice

1. INTRODUCCIÓN

Este documento contiene el manual de uso del componente visual ***Lista de valores*** del Framework Atlas. En él se incluye información sobre cómo utilizar dicho componente en una aplicación Web, así como información acerca de la configuración de los parámetros fundamentales del componente.

1.1. AUDIENCIA OBJETIVO

Este documento está orientado a toda aquella persona que esté desarrollando una aplicación Web basada en el Framework Atlas y necesite utilizar componentes de presentación en su aplicación Web.

1.2. CONOCIMIENTOS PREVIOS

Para un completo entendimiento del documento, el lector deberá tener conocimientos previos sobre las siguientes tecnologías:

- Java Server Faces (JSF)
- Facelets
- Richfaces
- Spring Framework
- Hibernate

Para saber más sobre dichas tecnologías, consultar el apartado de este documento, *Enlaces Relacionados*.

2. DESCRIPCIÓN

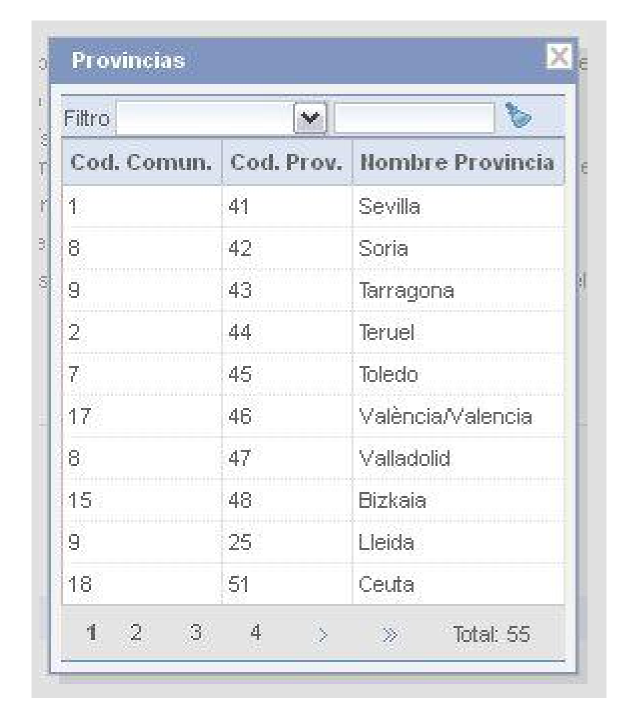
Este componente sirve para rellenar un campo de formulario con un valor elegido de entre un conjunto de múltiples valores. El usuario tendrá que pulsar sobre el botón del componente que deberá situarse junto al campo que se quiere rellenar. El componente tiene las siguientes características:

- La lista de valores se puede prefiltrar de manera opcional y previamente a que aparezcan los valores para elegir en el panel emergente, mediante controles que se dispongan para tal fin en el formulario, conectándolos a atributos de la etiqueta *listaValores*.
- El componente tiene 3 modos de funcionamiento:
    - Sin campo de texto asociado: sólo mostrará los botones de abrir panel y borrar elemento seleccionado.
    - Con campo de texto asociado: el componente mostrará un campo de texto para el registro seleccionado.
    - Con campo de tipo suggestion: se añade un campo en el que el usuario podrá escribir y

se sugerirán resultados. En este caso se podrán seleccionar registros sin necesidad de abrir el panel.



- El panel con los valores aparecerá solamente cuando el usuario pulse el botón correspondiente.
- El panel desaparecerá cuando el usuario seleccione un valor de la lista. También puede borrar el elemento elegido en el formulario si pulsa sobre el icono correspondiente. O bien cerrar el panel sin elegir ningún valor.
- Es posible ordenar la lista de valores por columnas.
- En la lista de valores aparecerá el número total de registros encontrados.
- El número de filas de la página se establece por un atributo en el componente.
- El componente utiliza ajax tanto para mostrar el panel y ocultarlo, seleccionar el registro y paginar, por lo que en ningún momento se realiza una recarga completa de la página. También es posible especificar qué otros componentes deben actualizarse al seleccionar un elemento o borrar uno seleccionado previamente.
- En la sección superior del panel aparecerá un campo y un desplegable para el filtrado de resultados. En el campo se establecerá el valor por el que se quiere filtrar y en el desplegable se secciona la columna por la cuál se quiere filtrar por el valor introducido en el campo. El filtrado puede ser automático cuando el usuario escriba en el campo de texto o cambie el valor del combo o bien con un botón para ejecutar la consulta.



Cod. Comun.	Cod. Prov.	Nombre Provincia
1	41	Sevilla
8	42	Soria
9	43	Tarragona
2	44	Teruel
7	45	Toledo
17	46	València/Valencia
8	47	Valladolid
15	48	Bizkaia
9	25	Lleida
18	51	Ceuta

3. INSTALACIÓN Y CONFIGURACIÓN

En este apartado se incluye información sobre la instalación y la configuración del componente Lista de Valores.

3.1. INSTALACIÓN

El componente de listado de valores ya viene instalado en el arquetipo Web, incluido con el módulo de componentes visuales. Por este motivo no es necesaria una instalación adicional si se parte del arquetipo.

3.2. CONFIGURACIÓN

Para la configuración del componente se debe crear una sentencia de consulta para la lista de valores.

Este paso es necesario para que el componente tenga acceso a los datos de la base de datos que queremos que sean mostrados en la lista.

Tenemos que establecer en el fichero queries.properties de la aplicación para cada consulta las siguientes dos propiedades, la primera es la consulta que se utilizará para recuperar los datos del listado de valores, la segunda propiedad es una query que devuelva el número total de registros.

El nombre de los parámetros debe componerse de la siguiente manera y deben ser únicos en el fichero de propiedades:

queries.properties

```
queryCode.nombreQuery=SELECT CAMPO1, CAMPO2 .... FROM ... WHERE ...

queryCode.nombreQuery.count= SELECT COUNT(1) FROM ... WHERE ...
```

El texto en color rojo es el identificador de la consulta. Es el valor que es necesario poner en el atributo *queryCode* cuando se establece un elemento *listaValores* en la página, por el cuál se puede seleccionar esta consulta cuando se inserta la etiqueta de este componente en un formulario.

Para el caso en que los datos no se deban obtener de una consulta sql el componente tiene un modo de funcionamiento en el que delega la obtención y el filtrado de los datos a unos métodos creados por el usuario, del mismo modo que se hace en la lista paginada, en este caso no es necesario crear las consulta en el fichero queries.properties. Este modo de funcionamiento se detalla en el siguiente apartado.

4. USO

Una vez instalado el módulo de componentes puede procederse a su utilización. El componente tiene dos posible orígenes de datos:

- Obtención de los datos mediante una consulta SQL.
- Obtención de los datos mediante métodos externos.

En el primer caso el componente se encarga de gestionar el filtrado y la ordenación mediante los atributos del propio componente y los del componente columnaListaValores. En el segundo caso se delega la responsabilidad en los métodos proporcionados por el usuario.

4.1. Consulta SQL

En este caso los pasos a seguir son los siguientes:

4.1.1. Paso 1: Definición de la consulta de la lista de valores

Es necesario especificar en el fichero queries.properties de la aplicación la consulta que tiene que usarse por la lista de valores que pongamos en nuestro formulario. Por ejemplo:

queries.properties

```
#  queries para la lista de valores de viales

queryCode.vialesSuca=
                  SELECT CDTVIA,DSTVIA,DSTVABR,ITNIVEL_USO
                  FROM DBA_SUCA.SUCA_TIPO_VIAL

queryCode.vialesSuca.count=
                  SELECT COUNT(*) FROM DBA_SUCA.SUCA_TIPO_VIAL
```

En este ejemplo se define una lista de valores para selecciona un tipo de vial.

4.1.2. Paso 2: Declaración en el bean de respaldo de la página de un objeto de tipo atlas.componente.bo.ListaValores e inicialización del mismo.

El bean de respaldo de la página debe definirse en el fichero de configuración de JSF que puede encontrarse en el proyecto web generado por el arquetipo bajo este nombre: *\src\main\webapp\WEB-INF\faces-managed-beans.xml.*


faces-managed-beans.xml

```
<managed-bean>
            <description>
                  Bean de respaldo de los ejemplo de Listas de
valores del Atlas Framework 1.0
            </description>
            <managed-bean-name>
                listaValoresTvialSampleBean
            </managed-bean-name>
            <managed-bean-class>
                atlas.samples.jsf.ListaValoresTvialSampleBean
            </managed-bean-class>
            <managed-bean-scope>request</managed-bean-scope>
            <managed-property>
                  <property-name>atlasFacade</property-name>
                  <value>#{atlasFacade}</value>
            </managed-property>
      </managed-bean>
```

En dicho bean debe crearse un objeto de tipo atlas.componente.bo.ListaValores e inicializarlo. Por cada lista de valores será necesario crear un objeto de este tipo.

ListaValoresTvialSampleBean.java

```
/**
 * The Class ListaValoresTvialSampleBean.
 */
public class ListaValoresTvialSampleBean extends BaseBean {

    ...


    /**
     * Lista de valores
     */
    private ListaValores listaValores;

    /**
     * (non-Javadoc)
     * {@inheritDoc}
     * @see atlas.faces.bean.BaseBean#init()
     */
    public void init() {
        this.listaValores = new ListaValores();
    }

    /**
     * @param listaValores
     */
    public void setListaValores (ListaValores listaValores) {
        this.listaValores = listaValores;
    }

    /**
     * @return la lista de valores
     */
    public ListaValores getListaValores () {
        return listaValores;
    }

    ...

}
```

4.1.3. Paso 3: Definición del espacio de nombres de etiquetas de Atlas

Es necesario crear un fichero *xhtml* y establecer la definición del espacio de nombres para las etiquetas de componentes de Atlas. Un ejemplo de cabecera de fichero *xhtml* es la siguiente:

Cabecera de fichero xhtml

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core"
      xmlns:ui="http://java.sun.com/jsf/facelets"
      xmlns:atlas="http://atlas.core.componentes/jsf"
      xmlns:a4j="http://richfaces.org/a4j">
```

4.1.4. Paso 4: Inserción en la página de la etiqueta atlas:listaValores

Es necesario insertar la etiqueta listaValores en el formulario de una página.

Un ejemplo de una lista de valores es el siguiente:

Ejemplo.xhtml

```
<atlas:listaValores id="listaValoresTvial" queryCode="vialesSuca"
aliasBean="listaValoresTvialSampleBean.listaValores"
dataModel="#{listaValoresTvialSampleBean.listavalores}"
titulo="Tipos de viales" rows="4" render="nombreVial">

  <atlas:columnaListaValores for="listaValoresTvial" campoBD="CDTVIA"
dataModel="#{listaValoresTvialSampleBean.listaValores}"
mostrarColumna="false" mostrarEnFiltro="false"
metodoRetorno="miManagedBean.setCodigoVial"/>

    <atlas:columnaListaValores campoBD="DSTVABR"
dataModel="#{listaValoresTvialSampleBean.listaValores}"
label="Abreviatura" render="nombreVial"/>

    <atlas:columnaListaValores campoBD="DSTVIA"
dataModel="#{listaValoresTvialSampleBean.listaValores}"
condicion=" LIKE 'C%'" label="Descripción" metodoRetorno="
miManagedBean.setNombreVial" render="nombreVial"/>

</atlas:listaValores>
```

4.2. Métodos externos.

En este caso el componente delega en dos métodos la obtención de los datos. Respecto al caso anterior no hay que definir ninguna consulta en el fichero queries.properties. El resto de pasos: declaración en el bean de respaldo del objeto ListaValores, definición del espacio de nombres de Atlas e inserción de la etiqueta son los mismos. Además hay que especificar los siguientes atributos en el componente:

- externalProvider: Debe ser true.
- pageProvider: Método al que se llamará para obtener la lista de elementos que se van a mostrar en la lista de valores. Se detalla a continuación.
- pageCountProvider: Método al que se llamará para obtener el número total de registros.

4.2.1. pageProvider.

El método pageProvider devuelve los registros que se muestran en una página de la lista de valores.

Ejemplo del método para el atributo pageProvider

```
    /**
     * @param index Indice de paginacion actual.
     * @param pageSize Tamaño de pagina.
     * @param order Criterio de ordenacion.
     * @param filter Filtro de busqueda.
     * @return Lista de objectos AtlasHashMap que representa la lista de registros.
     */
    public List<AtlasHashMap> obtenerEstadosCiviles(int index, int pageSize, Object
order, Object filter) {

        List<List<Object>> listaEstadosCiviles = new ArrayList<List<Object>>();

        // Obtención de la lista
        // ...
        //

        return this.listaValoresEstadoCivil.getPageProvider(listaEstadosCiviles);
    }
```

El primer parámetro es el primer elemento que se pide de la lista. El segundo el número de registros por página.

El método debe devolver una lista de objetos *atlas.componente.utiles.AtlasHashMap*. Para obtener esta lista se debe utilizar el método *getPageProvider* del objeto *ListaValores* asociado al componente tal y como se muestra en el ejemplo. A este método se le debe pasar una lista en la que cada elemento debe ser un registro de la lista. El formato del registro debe ser otra lista con tantos elementos como columnas tenga el componente en el orden en que se defina en la página xhtml. Y en esa lista cada posición será un valor a mostrar. En el siguiente ejemplo se muestra un ejemplo para estados civiles, la primera columna representa el identificador y la segunda la descripción:

1	Soltero
2	Casado
3	Viudo
4	Divorciado

En el parámetro order se recibe un *Object* con la ordenación, en este caso se puede hacer un casting a una clase de tipo *atlas.componentes.bo.OrderBean* que contiene la columna y el sentido de la ordenación en caso en que la hubiera.

En el último parámetro filter se recibe una lista con dos elementos, el primero es el valor del desplegable del filtro y el segundo el valor del campo de texto para especificar un filtro. Si alguno de los dos es nulo se recibe *null*.

4.2.1. pageCountProvider.

Este método debe devolver un *int* con el número de registros totales. Recibe como parámetro el objeto filter.

4.3. Atributos del componente.

Los principales atributos del componente atlas:listaValores son los siguientes:

- El atributo *id* sirve para establecer el identificador del componente. Es obligatorio y debe ser único.
- El atributo *dataModel* sirve para enlazar el componente con el objeto de la clase atlas.componentes.bo.ListaValores declarado.
- El atributo *aliasBean* identificar la ruta de acceso en el contexto de JSF del objeto anterior (Generalmente se corresponderá con la propiedad dataModel pero en modo Texto en lugar de como expresión).
- El atributo *queryCode* sirve para especificar cuál es la consulta a base de datos que se quiere utilizar para obtener los elementos que la lista de valores va a presentar. En este ejemplo se puede observar que este valor de identificador coincide con el mismo que se estableció como ejemplo en la sentencia de consulta de la lista de valores, de este documento.
- El atributo *titulo* sirve para establecer el título del panel emergente donde se presentan los elementos de la lista de valores.
- El atributo *rows* sirve para especificar el número de registros que se van a mostrar por página.

Los atributos de los componentes atlas:columnaListaValores son los siguientes:

- El atributo *for* debe ser el id de la lista de valores en la que esté el componente.
- El atributo *campoBD* identifica un campo o alias contenido en la clausula WHERE de la query de datos.
- El atributo *dataModel* sirve para enlazar con el objeto de la clase atlas.componentes.bo.ListaValores, debe corresponderse con el atributo *dataModel* del componente atlas:listaValores.
- El atributo *mostrarColumna* Indica si se debe mostrar el campo en el listado de resultados de la consulta, por defecto este atributo vale TRUE, en el ejemplo no mostraremos la columna CDTVIA
- El atributo *mostrarEnFiltro* Indica si se debe mostrar el campo en el combo desplegable de filtro de datos, por defecto el atributo vale TRUE. En el ejemplo no mostraremos la columna CDTVIA.
- El atributo *metodoRetorno* establece que cuando se seleccione un registro se debe ejecutar el método especificado pasando como parámetro el valor del campo para ese registro. En el ejemplo se llamará a los métodos **public void setCodigoVial(String codigoVial)** y **public void setNombreVial(String nombreVial)** de un managedBean llamado **miManagedBean** pasando como parámetros los valores de los campos CDTVIA y DSTVIA respectivamente paraun registro seleccionado.
- El atributo *inputRenderMode* especifica si se quiere mostrar un campo de texto asociado al componente y el modo de uso de este campo, los posibles valores son: none (no se muestra ningún campo, opción por defecto), readOnly (se muestra un campo de solo lectura) y suggestion (se muestra un campo en el que se podrán buscar y seleccionar registros).
- El atributo *inputColumn* debe especificarse si *inputRenderMode* es readOnly o suggestion, y debe ser el valor *campoBD* de alguna de las columnas que contenga la lista. El valor de esa columna será el que se muestre en el campo y sobre esa columna se buscará para sugerir resultados.
- El atributo *label* indica el título que debe utilizarse como cabecera en la columna del listado de valores y/o como descripción del item correspondiente en el combo desplegable de filtrado de datos. En el ejemplo los campos DSTVABR y DSTVIA se mostrarán al usuario como "Abreviatura" y "Descripción" respectivamente.
- El campo *condicion* permite la adición de condiciones adicionales a la clausula WHERE de la consulta sobre el campo especificado en campoBD. En el ejemplo se limita a la consulta a valores del campo DSTVIA que comiencen por C, añadiendose a la clausula WHERE la condición *"... AND DSTVIA LIKE 'C%' ..."*
- El campo *aliasTabla* permite especificar el alias de la tabla (en caso de que tenga) a la que pertenece la columna en la consulta que contenida en el queries.properties. Se utilizará para que en los filtros y condiciones se anteponga al nombre de la columna y así evitar posibles ambigüedades si varias tablas contienen una columna con el mismo nombre.
- El atributo *render* contendrá una lista de identificadores de los componentes que tengan que actualizarse al seleccionar un registro de la columna. Los identificadores deben estar separados por espacios.

La lista completa de atributos del componente listaValores son los siguientes:

Nombre atributo	Obligatorio	Descripción
Id	SI	Identificador del componente.
dataModel	SI	Modelo de datos empleado para una lista de valores. Sirve para extraer los datos que se presentan en la lista de valores. Se trata de un objeto de la clase atlas.componentes.bo.ListaValores que se encuentra declarado e inicializado en un bean de respaldo.
aliasBean	SI	Ruta de acceso en el contexto de JSF del objeto ListaValores que dará soporte al componente (Generalmente se corresponderá con la propiedad dataModel pero en modo Texto en lugar de como expresión).
queryCode	NO	Código de las consultas SQL necesarias para los datos y el paginado de la lista de valores, debe corresponderse con el distintivo de dos propiedades del fichero application.properties de la aplicación con formato queryCode.valorDeEstaPropiedad para la query de datos y queryCode.valorDeEstaPropiedad.count para la query que retorna el número de registros. Si se activa la propiedad externalProvider no es obligatorio.
externalProvider	NO	Valores true/false. Si es true no se utilizará la consulta sql del fichero queries.properties. En su lugar se obtendrán los elementos de la lista de los métodos señalados en las propiedades pageProvider y pageCountProvider. Su valor por defecto es false.
pageProvider	NO	Método de backingbean para obtener los elementos de la página. Se debe especificar si externalProvider es true.
pageCountProvider	NO	Método de backingbean para obtener el total de elementos existentes entre todas las páginas. Se debe especificar si externalProvider es true.
valorFiltro	NO	Valor que debe aplicarse en un filtrado inicial de datos, debe usarse en conjunto con la propiedad campoFiltro.
campoFiltro	NO	Campo sobre el que debe realizarse un filtrado inicial de datos, debe usarse en conjunto con la propiedad valorFiltro.
inputRenderMode	NO	Especifica si se quiere mostrar un campo de texto asociado al componente y el modo de uso de este campo, los posibles valores son: none (no se muestra ningún campo, opción por defecto), readOnly (se muestra un campo de solo lectura) y suggestion (se muestra un campo en el que se podrán buscar y seleccionar registros).

inputColumn	NO	Debe especificarse si *inputRenderMode* es readOnly o suggestion, y debe ser el valor *campoBD* de alguna de las columnas que contenga la lista. El valor de esa columna será el que se muestre en el campo y sobre esa columna se buscará para sugerir resultados.
suggestionSize	NO	Número de resultados sugeridos para el modo suggestion. Su valor por defecto es 10.
inputSize	NO	Tamaño del inputText para el caso *inputRenderMode readOnly*.
inputWidth	NO	Clase css para especificar el tamaño del input en el caso *inputRenderMode suggestion.*
inputClass	NO	Clase css para el campo input en los casos *suggestion* y *readOn ly*.
width	NO	Ancho de la ventana modal.
heigth	NO	Alto de la ventana modal.
moveable	NO	Valores true|false. Indica si la venta admite ser arrastrada con el ratón (Valor por defecto true).
resizeable	NO	Valores true|false. Indica si la venta admite ser redimensionada con el ratón (Valor por defecto false).
autosized	NO	Valores true|false. Indica si la venta debe autodimensionarse (Valor por defecto true). **Este atributo no funciona correctamente en Internet Explorer en cuanto al ancho, de modo que habrá que usarlo en combinación con la propiedad width para establecer el ancho correcto del panel modal. En este caso si no se especifica un ancho será 300px.**
titulo	NO	Título mostrado en la cabecera de la ventana.
msg_cerrar	NO	Mensaje emergerte correspondiente al botón de cerrar.
msg_filtro	NO	Mensaje mostrado como label del componente de filtrado.
msg_filtrar	NO	Mensaje emergente correspondiente al botón de filtrar.
msg_boton	NO	Texto del botón que abre la ventana modal.
render	NO	Id[‘s] (en formato compatible con UIComponent.findComponent()) de los componentes que deberán renderizarse al borrar el registro seleccionado en la lista. Puede ser un id, varios separados por espacio o una expresión EL que devuelva un array o Collection.
paginatorMaxPages	NO	Número máximo de links de cambio de página que se mostrarán simultaneamente en el paginado de la tabla de valores.
rows	NO	Número máximo de registros que se mostrarán por página en la tabla de valores.

filterInputSize	NO	Tamaño del input filtro.
filtroAutomatico	NO	Si es true el filtrado se hará automáticamente, en cada pulsación de una tecla o cambio en el valor del campo por el que se filtra se hará la búsqueda. Si es false habrá un icono para pulsar y realizar la búsqueda. Su valor por defecto es true.
Icono	NO	Ruta para el icono que mostrará el componente para abrir la lista.
Icono_limpiar	NO	Ruta para el icono que mostrará el componente para borrar el valor seleccionado.

Los atributos del componente atlas:columnaListaValores son los siguientes:

Nombre atributo	Obligatorio	Descripción
campoBD	SI	Campo o alias contenido en la clasula SELECT de la consulta que retornar los resultados sobre el que se aplicarán las propiedades de este componente
dataModel	SI	Referencia al objeto de tipo ListaValores que dará soporte al componente, debe corresponderse con la propiedad dataModel del componente listaPaginada
for	NO	Id del componente lista de valores que incluye la columna.
label	NO	Label que se mostrará como cabecera de la columna si es mostrada y del desplegable del filtro si el campo es usado en este (El valor por defecto es igual al valor de la propiedad campoBD)
condicion	NO	Condición en formato SQL que se aplicará al campo. Ejemplo: condicion=" LIKE '%texto%'" .
mostrarEnFiltro	NO	Valores true|false. Indica si se debe mostrar el campo en el combo del filtro de datos (valor por defecto true).
mostrarColumna	NO	Valores true|false. Indica si se debe mostrar el campo camo columna del listado (valor por defecto true).
metodoRetorno	NO	Indica un método de retorno al que se llamará pasándole el valor de este campo para el registro seleccionado, el valor debe tener el formato nombreBean.nombreMetodo y el método debe aceptar un String como parámetro de entrada y no retornar valor alguno.
aliasTabla	NO	Alias de la tabla que contiene la columna en la consulta que se utilizará para obtener la lista de resultados.
render	NO	Id['s] (en formato compatible con UIComponent.findComponent()) de los componentes que deberán renderizarse al seleccionar un registro de la columna. Puede ser un id, varios separados por espacio o una expresión EL que devuelva un array o Collection. Lista de id's de componentes separados por espacio.

filtrarAcentos	NO	Valor true|false para especificar si se debe hacer una búsqueda que filtre los caracteres acentuados. Su valor por defecto es false. Este atributo no tiene efecto si externalProvider es true.

5. RECOMENDACIONES Y BUENAS PRÁCTICAS

Se recomienda que el número de filas mostradas por página sea pequeño para que no aparezca el scroll vertical; para ello definimos una variable en la invocación al componente. Dicha variable es rows.

5.1. EJEMPLO DE USO

Se puede ver varios ejemplos de dicho componente en la siguiente secuencia de navegación por la aplicación de demostración de los componentes:

Inicio > Formularios > Lista de Valores



6. PREGUNTAS MÁS FRECUENTES

En este apartado se incluyen una lista de preguntas más frecuentes sobre el componente.

Pregunta: ¿Dónde puedo encontrar información general sobre los componentes?

Respuesta: En la aplicación de demostración de los componentes del Framework Atlas

Pregunta: ¿Cómo puedo crear un nuevo tipo de *Lista de valores*?

Respuesta: Para crear una nueva lista de valores lo primero necesario es establecer un nuevo grupo de elementos en el fichero de propiedades de la aplicación. En este fichero es necesario establecer un par de sentencias SQL encargadas de obtener los datos que aparecerán en la lista de valores (tanto para obtener los elementos en sí como el total de los elementos).

Pregunta: ¿Se ejecutan los métodos de retorno en caso de utilizar el modo suggestion?

Respuesta: Sí, seleccionar un elemento mediante suggestion genera los mismo resultados que seleccionarlo en el panel modal.

Pregunta: ¿Cómo puedo encadenar listas de valores? (que una dependa de lo seleccionado en otra)

Respuesta: Usando un valor de retorno de la lista padre en el atributo condición de la columna correspondiente en la lista dependiente. Ejemplo:

Ejemplo

```
<atlas:listaValores queryCode="provinciasSuca" aliasBean="listaValoresProvincias"
dataModel="#{listaValoresProvincias}" id="provinciasLV" titulo="Provincias"
rows="10" filterInputSize="10" width="400" height="400" moveable="true"
resizeable="false" paginatorMaxPages="4">

    <atlas:columnaListaValores campoBD="CDPAIS"
dataModel="#{listaValoresProvincias}" condicion=" LIKE '%ES%'"
metodoRetorno="listaValoresProvinciasSampleBean.setCodigoPais"
mostrarColumna="false" mostrarEnFiltro="false"/>

<atlas:columnaListaValores campoBD="CDCOMU" dataModel="#{listaValoresProvincias}"
label="Cod. Comunidad"/>

<atlas:columnaListaValores campoBD="CDPROV" dataModel="#{listaValoresProvincias}"
label="Cod. Provincia"
metodoRetorno="listaValoresProvinciasSampleBean.setCodigoProvincia"/>

<atlas:columnaListaValores campoBD="DSPROV" dataModel="#{listaValoresProvincias}"
label="Nombre Provincia"
metodoRetorno="listaValoresProvinciasSampleBean.setNombreProvincia"/>

</atlas:listaValores>


<atlas:listaValores queryCode="localidadesSuca" aliasBean="listaValoresMunicipios"
dataModel="#{listaValoresMunicipios}" id="localidadesLV" titulo="Localidades"
rows="10" filterInputSize="10" width="400" height="400" moveable="true"
resizeable="false" paginatorMaxPages="4">

    <atlas:columnaListaValores campoBD="CDPAIS"
dataModel="#{listaValoresMunicipios}" condicion=" LIKE '%ES%'"
metodoRetorno="listaValoresProvinciasSampleBean.setCodigoPais"
mostrarColumna="false" mostrarEnFiltro="false"/>

<atlas:columnaListaValores campoBD="CDPROV" dataModel="#{listaValoresMunicipios}"
condicion="='#{listaValoresProvinciasSampleBean.codigoProvincia}'" label="Cod.
Provincia" mostrarEnFiltro="false"
metodoRetorno="listaValoresProvinciasSampleBean.setCodigoProvincia"/>

    <atlas:columnaListaValores campoBD="CDMUNI"
dataModel="#{listaValoresMunicipios}" label="Cod. Localidad"
metodoRetorno="listaValoresProvinciasSampleBean.setCodigoLocalidad"/>

    <atlas:columnaListaValores campoBD="DSMUNI"
dataModel="#{listaValoresMunicipios}" label="Nombre Localidad"
metodoRetorno="listaValoresProvinciasSampleBean.setNombreLocalidad"/>

</atlas:listaValores>
```

7. ENLACES RELACIONADOS

Producto	URL
Ajax4JSF	http://www.jboss.org/jbossrichfaces/
Barbecue	http://barbecue.sourceforge.net/
Commons BeanUtils	commons.apache.org/beanutils/
Commons Configurations	http://commons.apache.org/configuration/
Facelets	https://facelets.dev.java.net/
Hibernate	http://www.hibernate.org/
Hibernate Annotations	http://www.hibernate.org/hib_docs/annotations/reference/en/html_single/
JAXB	http://java.sun.com/webservices/jaxb/
Jcaptcha	jcaptcha.sourceforge.net/
JPA	http://java.sun.com/developer/technicalArticles/J2EE/jpa/
JSF	http://java.sun.com/javaee/javaserverfaces/
JSFUnit	http://www.jboss.org/jsfunit/
Log4J	http://logging.apache.org/log4j/
MyFaces Core	http://myfaces.apache.org/
RichFaces	http://www.jboss.org/jbossrichfaces/
Spring	http://www.springframework.org/
Spring Security	http://www.springframework.org/
Velocity	http://velocity.apache.org/